Web-based honeypot for detecting and tracking attackers

Prashant Patil, Nisarg Thakur, Rajat Varade, Abhishek Pawar.
Department of Information Technology
SERS College of Engineering
Kopargaon, Maharashtra, India

ABSTRACT

Recently due to advancement of new technologies, various types of attacks on websites and webservers have also increased, some of them majorly include cross-site scripting (XSS) and SQL injection attacks which are used to spoil the vulnerability of the system and steal confidential data of the users like their user ID's and passwords from the server databases. Many methods were proposed to prevent such attacks. Some of them were created to learn about pattern of attacks and behavior of the attacker. That is honeypot. Honeypot is classified into two types based on the simulation that honeypot does, low interaction and high interaction.
In this project a low-interaction honeypot is proposed for emulating vulnerabilities that can be exploited by XSS and SQL injection attacks. This honeypot not only records attacker's request, but it also tries to expose attacker's identity by using browser exploitation techniques. Some have methods to hide their identity, thus they couldn't be tracked. Hence the proposed honeypot tries to overcome this problem by sending attackers a malicious JavaScript code. This JavaScript code will be run when an attacker opens the honeypot's website. The code steal attackers' confidential information like social account information by using Like Jacking technique, their IP and MAC addresses although even if they have used proxy or TOR to hide their identity. These information helps us to locate the attacker geographically by using some in-built packages and software that takes IP or MAC addresses as the input and provide its location as an output.

Keyword: - *Web-based Honeypot, SQL Injection, Cross-Site Scripting, LikeJacking*

I. INTRODUCTION

Web applications and web site shave become the main target of attacks. A survey conducted by Open Web Application Security Project (OW ASP) for finding the common attacks aimed at web applications. Some top attacks recorded were XSS and SQL injection. SQL injection is performed by exploiting vulnerabilities in the web applications that do not perform validation of the data inputted. Such vulnerabilities forces some parties to initiate the creation of a system that is specifically designed to record the patterns of the attacks and observe the behavior of attacker. The system hence is known as a honeypot.

A honeypot is a system that is created to emulate service that runs on a server to observe the pattern of attacks. In general, honeypot is divided into two types based on the level of interaction with attacker, namely high-interaction and low interaction honey pot. Low-interaction honeypot has a limited level of interaction because it only emulates a particular service on a system. While high -interaction honeypot has a high level of interaction because it uses the actual systems and services to be accessed by attackers.

Here a low-interaction, web based honeypot is built. It emulates XSS and SQL injection attacks that are most commonly done on web-applications. In addition, it will also collect attacker's information using a JavaScript code. If the request to honeypot is a normal HTTP request, honeypot will serve the request normally. However, if there is an indication of threat, honeypot will then simulate these attacks and send the response as if the attack is succeeded. For every request sent by attacker's browser, honeypot system will insert JavaScript codes into the response. These codes will be executed by the attacker's browser to collect certain information and send it back to the honey pot.

II. RELATED WORK

SQL Injection: Several methods have been developed to prevent the SQL Injection Attack. PHP Magic Quotes is a method to prevent SQL injection offered by PHP web framework [4]. It works by detecting specific character. If one or more of four specific characters, such as ', ", /, and NULL are found in POST, GET, and COOKIES data, \ character will be added in front of these characters. Wasserman [5] used static analysis combined together with automatic consideration. This method assumes that there is no tautology results from dynamically generated SQL query. Therefore, this method is very efficient in detecting SQL injection attack, but not for other attacks. Shin [6] had a method to create trial input data to find vulnerability of SQL injection by creating white-box on input flow and validation analysis.

A method to increase system's capability to detect and prevent SQL injection attacks based techniques like removal of SQL query attribute is needed. This paper talks about enhancing the detection of SQL Injection and gives its various enhancement techniques .Raspberry Pi is used in this paper as it is cheap and effective to be used as a honeypot. Raspberry Pi has a limited computational ability, hence a cluster is made to improve its power so as to get 64% accuracy in detecting SQL injection attacks, techniques to improve the detection of SQL Injection attacks. These include removal of SQL query attributes. Algorithm for the same for enhancement for detecting of SQL Injection attack is further described.

A log management server is essential to periodically collect log files from the attacker. The geographical location of the attackers can be shown on the world-map by using the WHOIS database and GeoPlot software. A WHOIS system is originated to find the originated country. GeoPlot software is used to create a geographical image of the dataset while the logging will help to prevent the further attacks from the same source and gather information about the nature of the attacks

III. PROPOSED SYSTEM

The main point of the proposed system is building a web-based low interaction honeypot that can bite. Not only record attacker's request, but also try to expose attacker's identity at the same time and prevent any further attacks from the same source by blocking its IP or MAC address and locating him geographically. The proposed system is aiming to classify the user by the request it sends to the server. If the user is a normal user it will be served normally. If it is an attacker, then send an emulated webpage pretending as a real webpage. Serve attackers attacks so as to give an effect as the attack is successful and send the desired output with attacked JavaScript.

The JavaScript runs on attacker's browser and sends the information like IP, MAC address, and attacker's social media account credentials. The information obtained is logged and the IP address is blocked for the further prevention of the attack. Using software's like GeoPlot and the information the attacker's location is plotted geographically.

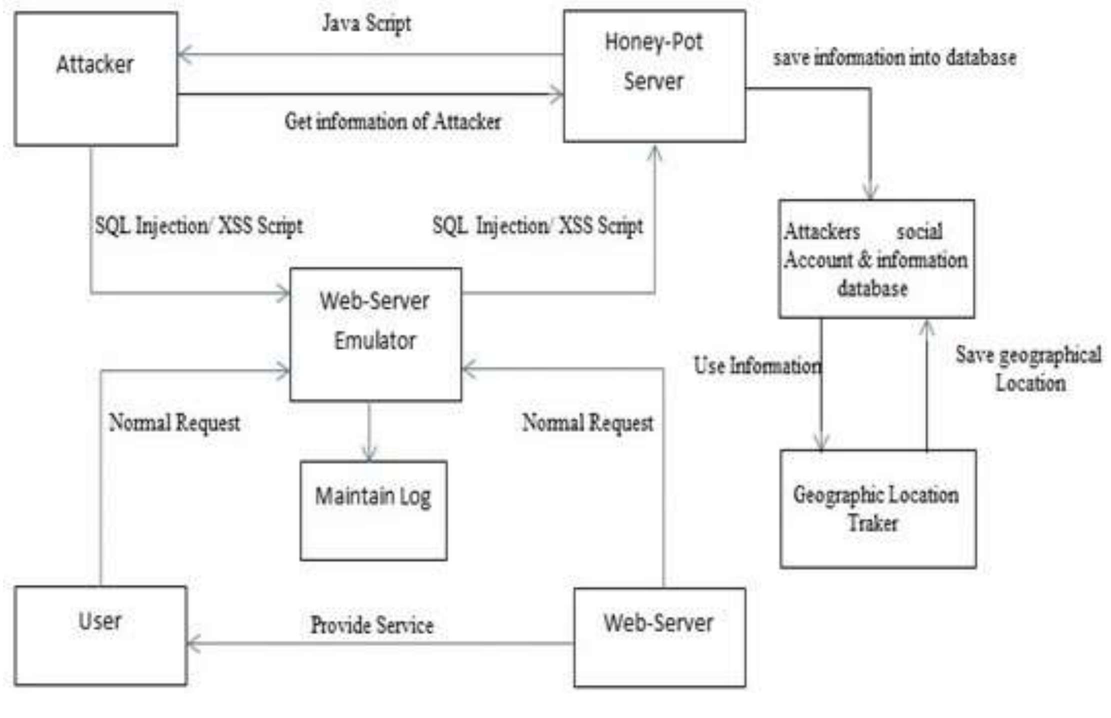
Attacker
Java Script
Honey-Pot Server
save information into database
Get information of Attacker
SQL Injection/ XSS Script
SQL Injection/ XSS Script
Attackers social Account & information database
Web-Server Emulator
Use Information
Save geographical Location
Normal Request
Normal Request
Maintain Log
Geographic Location Traker
User
Provide Service
Web-Server

Figure 1: Proposed System Architecture

The architecture of proposed system is composed of three main modules:

- Classification of Request.
- Emulator to serve attacker request and further get its information.
- Maintain the database of list of attackers and tracking their geographical location.

A. Classification of Request.

Classification of the request is needed to differentiate the attacker and the genuine user. It is done by inspecting the query entered by the user. Honeypot accepts the query, asses it and classifies as attacker's query or genuine request.

Attacker's query generally refers to the SQL Injection. SQL Injection The algorithm has a deletion mechanism of attribute value in a sent query by user. This algorithm needs two input data, fixed query (FQ) and dynamic query (DQ). FQ is a normal query run by a system with standard attribute value. This query is created by web-application developer when accessing to database. DQ is a query results from data inputted by user. The main point is FQ is figured as a query template, while DQ is a query result after changing attribute value from FQ by inserting attribute value from user. This method is very reliable to handle SQL injection.

Request from client will be examined by Honeypot server. If client request method is not POST, request will be redirected to web server. Otherwise, system will record sent data in POST parameter. The data will be filtered using SQL query attribute deleting method. If parameter passes the filter, it will be redirected to web server. If parameter cannot pass, then there is a SQL injection attack in the request parameter. For every detected request, it will be recorded as an attack. The system is also responsible for forwarding the reply from web server or honeypot to the client.

In more detail, the first step in proxy server is to parse received data and get request method from client. The proxy server should have capability to handle HTTP request, opening port 80, creating thread to handle request, and sending HTTP response.

```
Get HTTP request from client
Check request method
Get POST parameter data
Get FQ Create DQ based on POST parameter
Calculate FFQ and FDQ using f () function
XORing between FFQ and FDQ
If True:
    Forward HTTP request to honeypot cluster
Else:
    Forward HTTP request to real web server
Get response from honeypot cluster and send it to client
```

Figure 2: Classification of Request Algorithm

If request method is a POST, parameter will be processed as a dynamic query (DQ). It is used in SQL injection attack detection process. Figure. 2 briefly shows all process for each HTTP request. SQL injection detection is conducted in every HTTP request to proxy server. Received HTTP request will be managed to get attribute value. Afterwards, two queries are created. They are fixed query (FQ) which is normal query and dynamic query (DQ) which is query with attribute value based on user input. The next step is deleting attribute value in the two queries to get FFQ and FDQ.

```
f() function: -
 Status = 0
  For every character in query string
     If status is 0
          Add character to output
     If character is ‘ and status is 0
          Status = 1
     Else if character is ’ and status is 1
          Add character to output
          Status = 0
```

Figure 3: SQL query attribute deletion

Figure. 3 shows pseudo code for deleting attribute. This algorithm works by deleting query attribute value from FQ and DQ through f() function. Function f() is a function that implement algorithm to delete SQL query attribute value and will generates FFQ and FDQ. Decision whether a request is SQL injection attack or not is based on XOR operation between FFQ and FDQ which is the output from f(FQ) and f(DQ). The query that is resulted after deletion process (FDQ) will go through XOR operation with normal query that should be inserted by user (FFQ). If the result of XOR operation has FALSE value, it means the query that is inserted by user is a normal query that will be passed to database server to get query result as expected. Otherwise, if XOR operation returned TRUE value, it means the query that was inserted by user is expected as SQL injection attack.

B. **Emulate query and get the atttacker's information**

When the query is classified as Attacker's query, the attacker's request is then forwarded to the emulator. To lure the attackers, XSS and SQL Injection vulnerability are emulated so that they will think that the web-page is vulnerable. XSS and SQL Injection attacks are chosen since they are most conducted attacks now a days. The detail of the proposed method are explained below, they are interface design, request classification (XSS, SQL injection, or others), social media account gathering, and data storage.

The fake interface is designed in order to attract attacker and make them think that the website is vulnerable. Unlike Glastopf that make use of Google Hack Database to create fake website page, we are targeting real life attacker who opened the website, not bots nor machines.

Overall the fake page is just designed as the real web-page as attacker must not suspect that he is attacking fake page not the real web-site. The main page only consists of several fake information and obfuscated JavaScript code. With this, we made the attacker to think that our honeypot was an institution news website.

To get the attacker's information, instead of Java Applet, as Java Applet is protected by most of new browser and cannot be used to expose attacker's identity, JavaScript is utilized. Just like Applet, JavaScript is run by the browser, so it can be used to access some of client's credentials.

The proposed honeypot makes use of LikeJacking technique which is usually used by black-hat advertisers. Initially a dummy Facebook Page is created. In LikeJacking, this dummy Facebook page is liked by the attacker accidentally when they visit the honeypot. There are two limitation of this method. Firstly, the attacker must be logged on to his Facebook account when they visit the honeypot. Secondly, the attacker could remove their 'like' from our Page anytime, so we have to rely on Page's Notification rather than List of Likes. Although they have left destined page, their account name is still listed on notification.

When the above technique fails, i.e if the attacker's Facebook credentials are not stored on his browser, then JavaScript is used for finding the IP address and MAC address of the attacker's system. The JavaScript will run on the user's machine and send back the IP address and MAC address of the attacker back to Honeypot.

In this way, we have either directly the Attacker's Facebook page information that makes us easy to track him, or we have the IP address and the MAC address of the attacker which can be further used in several packages to track its geographical location in the World.

C. **Tracking and Maintaing the Arrackers information**

The information retrieved back from the JavaScript is stored in a Database so that the attack from the same source can be prevented again.

The database is simple MySQL database which has information such as IP address, MAC address and Source Country and if available Facebook account User name and User ID.

The Geographical location of the attacker is found by the source IP address sent by the Java Script is forwarded to the WHOIS database to find the originated country.

The WHOIS system originated as a method that system administrators could use to look up information to contact other IP address or domain name administrators (almost like a "white pages"). The use of the data that is returned from query responses has evolved from those origins into a variety of uses including domain name, admin name, email address and post address of the admin, and country.

The results will be summarized based on country, stored in the database and plotted on the world map. For plotting on the world map, we utilize the Geoplot source code. GeoPlot [IS] is a light-weight java applet which allows users to create a geographical image of a data set. The input to the GeoPlot software is IP address of the attacker and the software shows the Geographical location of the attacker on the world map.

IV. CONCLUSION

Honeypots are prominent for learning behaviors of intruders and attackers. The project aims in maintaining the system integrity and protecting the confidential data from the malicious users or attackers. It could get much more data from attacker, in addition, it could also expose attacker's identity by doing LikeJacking technique. From the received data, we can see that some of the attackers were not protecting their identity, which make them easier to catch. Proposed system works by initially classifying the user by its request into either normal user or attacker. Information from Honeypot will be more accurate and more useful when we can observe from multiple Honeypot servers. This paper aims to create a prototype system of Honeypot log management server in order to facilitate automatically log transfers. Information retrieved are analysed and maintained on the MySQL database server. We also design and implement a web-based interface to access the data and summarize the results. Lastly, the geographical distribution of attackers is also plotted on the world map via Geoplot API, so that the administrators can learn the origin country of the attackers. For further work, we would like to support more types of service log files and visualize more data from the servers. Lastly, more experiments in the real network should be done in order to prove the effectiveness of the implementation.

REFERENCES

[1]. “Aggressive Web Application Honeypot for Exposing Attacker's Identity”, Supeno Djanali, FX Arunanto, Baskoro Adi Pratomo, Abdurrazak Baihaq Hudan Studiawan, Ary Mazharuddin Shiddiqi ,2014 1st International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE), 978-1-4799-6432-1/14/ ©2014 IEEE.

[2]. ”Distributed Honeypot Log Management and Visualization of Attacker Geographical Distribution”, Vasaka Visoottiviseth, Uttapo lJaralrungroj, Ekkachai Phoomrungraungsuk and Pongpak Kultanon, 2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), 978-1-4577-0687-5/11 ©2011 IEEE.

[3]. “Design of Distributed Honeypot System Based on Intrusion Tracking”, Yun Yang, Hongli Yang, JiaMi“Research on the application of honeypot technology in Intrusion Detection System”, XiangfengSuoa, XueHanb, YunhuiGaoc.

[4]. ”Sophisticated Honeypot Mechanism –the Autonomous Hybrid Solution for Enhancing.Computer System Security”, LiberiosVokorokos, Peter Fanfara, JánRadušovský and Peter Poór, 978-1-4673-5929-0/13/ ©2013 IEEE.

[5]. “SQL Injection Detection and Prevention Detection system using Raspberry Pi Honeypot Cluster for Trapping Attacker” Supeno Djanali, FX Arunanto, Baskoro Adi Pratomo, Hudan Studiawan, Satrio Gita Nugraha.

[6]. “Computer System Security”, Liberios Vokorokos, Peter Fanfara, Ján Radušovský and Peter Poór, 978-1-4673-5929-0/13/ ©2013 IEEE.

[7]. “Research on the application of honeypot technology in Intrusion Detection System”, Xiangfeng Suoa, Xue Hanb, Yunhui Gaoc.